BASIC8 UTILITIES

by M.D.Pelletier

Table of Contents

A Reference Manual for

BASIC8 Utilities

Utilities designed to aid BASIC Programming.

TRADEMARKS

Atari is the trademark of Atari, Inc., Sunnyvale, CA
OS/A+ is a trademark of Optimized Systems Software, Inc., Cupertino, CA

Preface

BASIC8 is a series of 23 utility programs intended to aid the programmer in entering and debugging Atari BASIC programs. The design is that the computer should serve as a tool, not a hinderance in debugging a program. BASIC8 is written entirely in machine language.

System Requirements

BASIC8 is intended to be used with the Atari BASIC ROM, A minimum of 32K (24K for cassette) is recommended. It uses about 7000 bytes of memory (6660 for cassette version), loads itself into low memory (MEMLO) and protects itself from BASIC and SYSTEM RESET. MEMLO (location 743 & 744) is reset to point just above BASIC8.

BASIC8 loads into memory starting at $4000 (page six of memory is used at load time, the cassette version loads directly to $700). MEMLO is then examined and BASIC8 is relocated in memory starting at the first page boundary above MEMLO. Thus if MEMLO is $1CFC (7420) at load time BASIC8 will relocate itself to start at $1D00 (7424) and will reset MEMLO to $384F (14415).

If you wish BASIC8 to load at a different location (other than default MEMLO) you may re-define MEMLO prior to loading BASIC8. For example if you type "POKE 743,0:POKE 744,48" followed by loading BASIC8 would result in BASIC8 starting at $3000 (48*256=12288). The highest MEMLO allowed is $4000 (16384), "POKE 743,0:POKE 744,64". Remember that MEMLO is always reset to just above BASIC8.

Using BASIC8

Atari DOS

Power up the disk, monitor and computer in the "normal" fashion. Type "DOS", when the menu comes up type "L" (to select the load binary file option) then type "BASIC8.COM" for the requested filename.

Alternately you can rename "BASIC8.COM" to "AUTORUN.SYS" in which case BASIC8 will load automatically at boot time.

OS/A+ DOS

Power up the disk, monitor and computer in the "normal" fashion with DOS booted. Type "DOS" then type "BASIC8" and BASIC8 will be loaded and activated.

Cassette Based System

Power up the monitor, hold down the START key and power up the computer. One audible 'beep' should be heard. Set up the cassette with BASIC8 and press the PLAY button on the cassette. Press the RETURN key on the computer and wait for BASIC8 to load and activate.

Overview of BASIC8

The commands can be broken down into four types:

1. BASIC Programming	ASM
	AUTO
	COPY
	DEL
	FIND
	RENUM
	SOFT
	VAR
2. Disk CIO Functions	DIR
	FORMAT
	LOCK
	PURGE
	RENAME
	UNLOCK
3. Memory/Disk (map,patch)	= (hex/dec)
	BLOAD
	BSAVE
	DISKPEEK
	DISKPOKE
	MEM
	STOR
4. GENERAL	HELP
	PSCREEN

BASIC8 gets its name from the eight commands listed above which are most useful for 'BASIC Programming'.

BASIC8 defines and uses an "M" driver to which it writes and examines data in memory.

Arguments, Syntax

The conventions used in this booklet are as follows:

1. Capital lettters denote commands, parameters etc. which must be typed in exactly as shown (i.e. DIR, VAR).

2. Lower case letters specify items which may be used (parameters, options etc.). These types of items are as follows:

number	number, 0 to 65535
begin	number, 0 to 32767, BASIC line number
end	number, 0 to 32767, BASIC line number
increment	number, 1 to 32767, between BASIC lines
filename	*see below
oldname	file-specifier, *see below
newname	file-specifier, *see below
start	number, 0 to 65535, memory address
stop	number, 0 to 65535, memory address
sector#	number, 1 to 720, disk sector number
disk dev#	number, 1 to 4
string	a set of characters (to search for)
device#	device letter, a number, a colon (ie. D1:)
Dn	(same as device#, except disk only)
startrow	number, 0 to 23, screen row
stoprow	number, 0 to 23, screen row
map	either number(s) or a string literal

3. Items in parentheses indicate optional items, ie. (,begin).

4. All numerics that are BASIC8 parameters may be entered as decimal OR hexadecimal numbers (inter-mixing with multiple numbers is allowed). Any number beginning with a "$" will be interpreted as hex otherwise the number will be decimal.

*The following convention is used for filenames within this booklet:

Filename = device letter, device number, a colon then a file-specifier

Example: D1:UTIL.COM
- device letter="D"
- device number="1", followed by a colon
- file-specifier="UTIL.COM"

Filenames usually refer to disk files, however cassette files are supported by BASIC8.

COMMAND PROCESSOR
Command Entry

Commands are typed in similar to BASIC direct commands (ie. RUN, LIST) with the following restrictions:

1. Commands must be left justified; no leading blanks.
2. Multiple commands (separated by semicolons) are not supported. Use single command statements.

Page Zero Sharing

Memory locations $CB through $D2 (unused by BASIC) are left unchanged by BASIC8 and page six is not used by BASIC8. This is of note to programmers who use machine language programs with BASIC.

Disable BASIC8

BASIC8 works by intercepting all data received by the screen editor and examining it to see if it is a BASIC8 command. It does not know if a BASIC program is running.

NOTE!
If your application program uses the screen editor for any data entry (ie. INPUT A$), BASIC8 will intercept it if the COMMAND PROCESSOR interprets it as a BASIC8 command.

After intercepting the screen editor data, BASIC8 first examines memory location 1008 ($3F0). If a zero value is found then BASIC8 continues, if a non-zero value is found than BASIC8 returns control to BASIC. You may disable BASIC8 in one of two ways:

1. From the keyboard type CTRL-SHIFT-ESC (hold down both the CTRL and the SHIFT buttons then strike the ESC key). This keystroke toggles between disable BASIC8 and enable BASIC8.
2. Programatically, to disable "POKE 1008,1", to enable "POKE 1008,0".

The BREAK key will function normally with Disk/Printer CIO functions however it will not work with several BASIC8 routines. BASIC8 sometimes disables the keyboard while processing. Some commands allow you to abort by pressing a consol key (ie. the START key).

NOTE!
BASIC8 is automatically disabled when you invoke the "DOS" command. It is not re-enabled upon return to BASIC, you must manually enable it.

CAUTION!
Do NOT enable BASIC8 while within the Atari DOS Menu program or it's ga-ga time (the system will hang).
If SYSTEM RESET is struck while in DOS or before enabling BASIC8 after returning from DOS then BASIC8 will be disabled and you will not be able to re-enable it (except by re-booting it).

SOFTKEYS

Every key (other than operating keys CTRL, SHIFT and BREAK) can be assigned a string of up to 120 characters. This booklet will limit the SOFTKEY discussion to the "number" keys (1,2,...,0), and the alpha keys (A,B,...,Z). The intention of SOFTKEYS is to have pre-defined assignments which improve programming speed an accuracy.

Defining SOFTKEYS

To define a SOFTKEY, type the ESC key then while holding down the START button type the key you wish to define. The current string assigned to that key (if any) will be displayed, followed by "Define Key#1" (in this case assume key 1 is to be defined). The computer is now waiting for you to type the new string assignment for the chosen SOFTKEY. You may use all of the screen editor EDIT capabilities. When the string looks good to you strike the RETURN key and the softkey is now defined. If you have a large BASIC program in memory this may take a few seconds. If you select to define a softkey but wish to abort type either the BREAK key or the CTRL-3 keystroke.
Note that SOFTKEYS are disabled while defining a SOFTKEY.

Using SOFTKEYS

To use a SOFTKEY, merely press the SOFTKEY you wish to use while holding down the START button. The string assigned to that SOFTKEY will be printed to the screen.

Display All SOFTKEYS

To display all SOFTKEYS currently defined type CTRL-SHIFT-<space> (hold both the CTRL and the SHIFT keys down then press the space bar). For each key currently defined the key name followed by one space and then the definition for that key will be displayed. You may abort the listing by pressing a consol key (ie. START key). The SOFTKEYS will be sequenced in order of their keycode value as from the POKEY chip (for instance numerics will sequence as follows: 4,3,6,5,2,1,9,0,7,8).

Suggested Use and Notes

I recommend using the numeric SOFTKEYS for current working strings which may change frequently. You can paste a piece of paper just above the keys and quickly jot down their current definitions. FOr the alpha keys you might assign most used commands/functions and leave them (for the most part) alone. You can save SOFTKEYS on file, and with continued use you'll know thier definitions by memory. Bear in mind that each byte used in a SOFTKEY definition uses one byte of free memory.

COMMAND	PARAMETERS	DESCRIPTION
=	number	displays numbers, "decimal,hex" form
ASM	--	enter mini assembler/disassembler
AUTO	(begin)(,increment)	executes auto line numbers (BASIC)
BLOAD	filename	binary load to memory of exist. file
BSAVE	filename, start, stop	bin. mem. save start->stop,to a file
COPY	begin, end, to, (,by)	copy BASIC lines->diff seq. location
DEL	begin (,end)	deletes BASIC program lines
DIR	(Dn)(:file-specifier)	displays disk directory information
DISKPEEK	sector#, (,disk dev#)	reads/displays given disk sector
DISKPOKE	sector#, (,disk dev#)	writes/displays given disk sector
FIND	" " (begin)(,end)(,1)	lists all lines with search string
FORMAT	device#	formats a disk
HELP	(type)	displays commands/functions avail.
LOCK	filename	protects an existing file on disk
MEM	(start)	displays memory in hex/alpha form
PSCREEN	(startrow)(,stoprow)	prints hardcopy of screen, GR.0
PURGE	filename	deletes an existing disk file
RENAME	Dn:oldname,newname	renames existing disk file(s)
RENUM	(begin)(,increment)	renumbers seq. lines of BASIC prog.
SOFT	filename(,Save)	loads/saves SOFTKEY data files
STOR	startaddress,map	stores numbers/strings to memory
UNLOCK	filename	unprotects an existing file on disk
VAR	--	displays all variables used in prog.

To DISABLE/ENABLE BASIC8 hold both CTRL and SHIFT and type ESC key
SOFTKEYS: use - hold START key and press the desired SOFTKEY
define - press ESC key, hold START and press desired key
display- hold both CTRL and SHIFT and press the space bar

Command : =

Purpose : display number in decimal and hexadecimal form

Form : =number

Argument : a decimal number (or hex if preceded with a "$")

Description

This command allows you to convert numbers quickly from decimal to hexadecimal (and vice-versa). Hexadecimal numbers must be preceded by a "$", there must be no spaces between the "$" and the following hexadecimal number.

```
Example: =$2E7
         743 , $02E7

     or: =743
         743 , $02E7
```

Command : ASM

Purpose : assembles & disassembles in memory

Form : ASM

Arguments : (none)

Description

The assembler/disassembler allows instant assembly of instructions directly to memory and immediate disassembly of instructions in memory. Normal usage provides flexible address prompting, you are not required to remain on the prompted line and all EDIT keys may be used normally. A simple labeling capability of up to 16 labels is provided. You may use a label for any operand in an instruction.

When you enter the assembler/disassembler:
1. The screen is cleared and reset to graphics mode 0.
2. The user is in the assembler/disassembler Command Interpreter.
3. The 'logical line' becomes one line (maximum of 38 characters).
4. All numerics are interpreted as 16 bit positive numbers 0 to FFFF, one to four hexadecimal digits. Overflow isn't detected or reported as an error. If a one byte numeric is expected then only the low byte is used
5. Standard MOS Technology mnemonics are used for opcodes. It is assumed that the user is familiar with assembly language

The assembler/disassembler Command Interpreter:
The first nonblank character is always the "command". Commands are one character long and there are only four valid commands in the assembler/disassembler:

Command	Name	Meaning
,	comma	assemble with next address prompt disabled
.	dot	assemble with next address prompt enabled
-	dash	disassemble with next address prompting
/	slant	exit to BASIC

DISASSEMBLE

To disassemble type a dash followed by the address to start disassembling and a RETURN. If you omit the address, disassembly will start at the program counter (PC) last address. Twenty instructions will be disassembled to the screen, the program will halt with the last line a dash followed by the next address to disassemble. At any time you may cursor up to an instruction and reassemble changes to the listing.
Example:

```
-600                (this will commence disassembly at hex 600)
,0600  LDX #4     ; A2 04
```

ASSEMBLE

To assemble type a dot followed by an address, an instruction and a RETURN. This will be assembled, and you will be prompted with the next address. At any time you may cursor up, EDIT and re-assemble an instruction. If you type only a dot and a RETURN, you will be prompted with the current PC. The 'form' of the entered line to assemble is as follows:

```
.ADDRESS (LABEL) OPCODE (OPERAND)     (ie  .0600  LDX #4)
```

Note the label is optional and the OPERAND is optional or as required by the addressing mode of the OPCODE.

There are three pseudo-ops that may be used while assembling:

```
***              this skips one byte, leaves memory unchanged
BYT  address     inserts 1 byte address at the current PC
ASC  literal     inserts (multi byte) literal at the current PC
```

Note, the immediate OPERAND (ie. #41) may be satisfied with an ATASCII character in the following form: #'char (pound sign, apostrophe, followed with the character). for example "LDX #'A" will assemble to "LDX #41".

LABELS

Labels are entered by enabling the inverse character key (the Atari key) and typing a single digit alpha A to P. Labels may be used anywhere (while assembling) that a numeric 1 or 2 byte address (or immediate value) is required in the OPERAND.
A label is resolved by typing that label in front of the OPCODE of an instruction. If a label remains unresolved when exiting the assemble mode, the user will be prompted to define the label. A valid entry will be required, to abort strike the break key. There are 16 valid labels allowed (inverse A to P), each label may have a maximum of 8 unresolved references.
The assembler uses addresses $3C0 to $47F to track unresolved label references.

ERRORS

Any error number less than 40 (while in the assembler) denotes the position along the entered line where the error was encountered (starting with position 0). any error number greater than or equal to 40 has the following meaning:

```
40   parenthesis not closed
41   address parameter required, but not found
42   branch address out of bounds
43   zero page indirect X or Y not specified
44   --(not used)--
45   maximum of 8 unresolved label references exceeded
46   too many parameters in OPERAND
47   bad instruction generated
```

Command : AUTO

Purpose : execute automatic line numbering for BASIC programing

Form : AUTO (begin) (,increment)

Arguments : begin - optional, starting line number for auto-numbering, default= last line + 10
increment- value between lines for auto-numbering, default= 10

Description

This command will enter the line numbers automatically. this is useful when typing in a new BASIC program.

After typing this command you are in 'Auto-Number' mode. When you type the first character for a new BASIC program statement you will see the line number printed to the screen followed by the character you just typed. the line number is not printed to the screen until after the first character of the new logical line is typed. If you type a 'RETURN' with nothing else keyed in you will exit the 'Auto-Number' mode.

NOTE

1. When you type this command you will be in 'Auto-Number' mode until you type a 'RETURN' at the beginning of a new logical line with nothing else keyed in.
2. EDIT keys may be used normally you are not required to remain on the line where the auto-number was printed
3. If you do not specify the 'begin,increment' the auto-numbering will begin at the last line+10 increment will be by 10's.
4. 'Auto-Number' mode is exited if:
   a. an existing statement is detected with the same line number as the next auto-number
   b. the next auto-number is an invalid line number (ie. greater than 32767)
   c. a return is typed at the beginning of a logical line

5. You may use SOFTKEYS normally but you will be unable to define SOFTKEYS while in the 'Auto-Number' mode. You must exit the 'Auto-Number' mode to define (or re-define a SOFTKEY). See SOFTKEYS on page 6 of this booklet.

Example: AUTO 100

Command : BLOAD

Purpose : loads existing binary files (memory image) to memory

Form : BLOAD filename

Argument : a filename

Description

Binary image files may be loaded directly into memory with this command. the files must be compatible with "normal" binary object files that is:

1. the first two bytes must be $FF, $FF
2. the next four bytes must be the start, stop addresses to load into memory
3. the remainder of the file is the binary image to be loaded

Binary files may be loaded from either disk or cassette. Only single step binary loads are supported. The number of bytes transferred and the starting address (in decimal , hex form) will be displayed.

Example: BLOAD DO.COM
Count=76 , address=16384 , $4000

Command : BSAVE

Purpose : saves a portion of memory to a file (disk or cassette)

Form : BSAVE filename, startaddress, stopaddress

Arguments : a filename
startaddress- to start save
stopaddress - to stop save

Description

A binary image of a portion of memory may be written to a file in standard object file format that is:

1. the first two bytes are $FF, $FF
2. the next four bytes are the start, stop addresses (as typed in by the user
3. the remainder of the file is the binary image of the memory being saved

Binary files may be saved to either disk or cassette.

Example: BASVE DO.COM $4000,$404C

Command : COPY

Purpose : copies BASIC program statement(s) to another sequence location

Form : COPY begin,end,to (,by)

Arguments : begin- begin range of BASIC statements to be copied
end - end range of BASIC statements to be copied
to - line number to start copied statements
by - optional, increment between copied lines (default= 1)

Description

The COPY command allows you to copy existing program statements to another sequence location. Line numbers to within the copied block of statements will be renumbered, all line number references to lines not in the copied block of statements will be ignored. The number of statements copied will be displayed upon completion. All parameters are required except the 'by', default 'by' equals one.

The COPY command can be used to move statements to different sequence location. A possible procedure to accomplish this is as follows:

1. RENUMber the program leaving enough 'space' between lines where you wish to move the statements
2. COPY the statements to the new sequence location
3. FIND and modify statements refering to lines in the old location
4. DELete the old copied statements
5. RENUMber the program

Example: COPY 110,140,200,2
Count=4

Command : DEL

Purpose : deletes BASIC program statement(s)

Form : DEL begin (,end)

Arguments : begin- begin range of BASIC statements to be deleted
end - optional, end range of BASIC statements to be deleted (default= begin)

Description

The DELete command allows you to delete BASIC program statements. A single line or a block of lines may be deleted.

The begin range must be an existing line number. The only requirement for the optional end range is that it be greater than or equal to the begin range.

Example: DEL 100,120
or: DEL 1020

Command : DIR

Purpose : displays the disk directory

Form : DIR (Dn:) (file-specifier)

Arguments : Dn: - optional, disk device name/number, default= "D1:"
file-specifier- optional, any valid file name which may contain 'wild-card' characters "?" and "*", default= "*.*"

Description

The DIRectory command displays all files on the current default disk (as limited by the optional file-specifier). The default device is modified if the given disk device name/number is specified. If the file-specifier is given, all files matching the specifier will be displayed to the screen. If you wish to select a file-specifier, then you ust also give the disk device name/number.

Example: DIR D1:*.BAS
or: DIR

Note: if the device number is omitted from the disk name/number, then the current default device is used (ie. DIR D:UTIL?.*)

Command : DISKPEEK

Purpose : to read and display a sector from disk

Form : DISKPEEK sector# (,disk dev#)

Arguments : sector# - disk sector number to read and display
disk dev#- optional, disk device number, default= 1

Description

The DISKPEEK command reads one 128 byte sector from disk device #1 (or as specified), then displays this inforamtion in hex and alpha form. This command always reads 128 bytes (one sector) and stores it at address $400 (1024).

This command may be used effectively with the STOR and DISKPOKE commands. If you are patching a file or information on disk you may DISKPEEK the sector, STOR new hex/alpha data then DISKPOKE the sector back to disk.

Note that the File Manager is by-passed with this command. 'Normal' checks and file search routines are not used. The disk I/O is handled by the resident disk handler in ROM. A single density disk is assumed.

Example: DISKPEEK $169

Command : DISKPOKE

Purpose : to write a sector to disk, then display the data

Form : DISKPOKE sector# (,disk dev#)

Arguments : sector# - disk sector number to read and display
disk dev#- optional, disk device number, default= 1

Description

The DISKPOKE command writes one 128 byte sector to disk device #1 (or as specified), then displays this inforamtion in hex and alpha form. This command always writes 128 bytes of data from computer memory starting at address $400 (1024) to the specified disk sector.

This command may be used effectively with the DISKPEEK and STOR commands. If you are patching a file or information on disk you may DISKPEEK the sector, STOR new hex/alpha data then DISKPOKE the sector back to disk.

Note that the File Manager is by-passed with this command. 'Normal' checks and file search routines are not used. The disk I/O is handled by the resident disk handler in ROM. A single density disk is assumed.

CAUTION!

No checks are performed to determine if data is overwriting a locked file, or even DOS.SYS! You could destroy the integrity of the disk if you incorrectly modify sector#360 through 368 (the VTOC and Directory). For this reason any attempt to invoke this command will prompt you with "Verify (Y/N)", to which you must respond "Y" if you wish to perform this command.

Example: DISKPOKE 720
Verify (Y/N)

Command : FIND

Purpose : to locate all occurances of a given set of characters

Form : FIND "string" (begin) (,end) (,list)

Arguments : "string"- a delimeter, a set of characters to search for, a delimeter
begin - optional, begin range of BASIC statements to be searched
end - optional, end range of BASIC statements to be searched
list - optional, "0" (zero) means list lines found, "1" (one) means list line numbers only, default= 0

Description

The FIND command locates and lists the lines for all occurances of the specified set of characters (as limited by begin, end ranges). Note that if the 'list' parameter is specified as "1" then only line numbers will be listed (not the whole line). The delimeter is required and the end delimeter must match the begin delimeter. Any character (except a blank) may be used as a string delimeter (ie. a quote or the slant are both valid delimiters).

The string may be generalized to include the use of the 'wildcard' character "*" (the "heart" symbolized here with an asterisk, hold the CTRL key and strike the comma key). This wildcard character in the string allows any character in that position to produce a match, for example FIND /A(*)/ will find both A(1) and A(2) .

To prematurly abort the FIND command, press either the BREAK key or one of the CONSOL keys (ie. the START button).

Example: FIND /INFO$/
or: FIND "MYPMBASE" 1000,2000,1

Command : FORMAT

Purpose : formats a disk

Form : FORMAT Dn

Argument : Dn - disk device name/number

Description

The FORMAT command formats a disk so that it may be read from or written to by programs.

CAUTION!

This command erases any previous data on a diskette. For this reason any attempt to invoke this command will prompt you with "Verify (Y/N)" to which you must respond "Y" if you wish to perform the command.

Example: FORMAT D1
Verify (Y/N)

Command : HELP

Purpose : displays available commands, functions or operators

Form : HELP (type)

Argument : type - optional, type of help (ie. B,C,F or O)
default= B

Description

The HELP command displays commands available with BASIC8. The standard BASIC commands, functions and operators may also be displayed.

The types of HELP are as follows:
1. B - BASIC8 commands
2. C - BASIC commands
3. F - BASIC functions
4. O - BASIC operators

Example: HELP C

Command : LOCK

Purpose : protect file(s) from erasure, writing or rename

Form : LOCK filename

Argument : a filename

Description

The LOCK command allows locking of all files matching the file-specifier on the named disk device. These files will then be shown with a preceding asterisk when a "DIR" command is issued.

Example: LOCK D1:UTIL.M65

Command : MEM

Purpose : displays the contents of 128 bytes of memory in hex and alpha form

Form : MEM (start)

Argument : start- optional, start address in memory, default=last MEM address+128 bytes

Description

The MEM command displays the contents of memory, 16 lines with 8 bytes per line for a total of 128 bytes. The form of each line displayed is the hex address of the first byte in the line, the hex contents of successive memory locations for 8 bytes. Then the ATASCII character interpretation of the positionally corresponding hex contents for 8 bytes. The form of the display is as follows:

```
addr .....hex-contents..... ATASCII.
F0F9 43 4F 4D 50 55 54 45 52 COMPUTER
```

When the MEM command is issued, you will remain in the "MEM mode" until the 'RETURN' key is typed. If the space bar is pushed (or any other key except RETURN) the next 128 bytes of memory will be displayed and you will still be in the "MEM mode". To exit the "MEM mode" push the RETURN key.

Example: MEM $F0F1

Command : PSCREEN

Purpose : dumps the contents of screen memory to the printer (GRAPHICS mode 0 only)

Form : PSCREEN (startrow) (,stoprow)

Arguments : startrow- optional, row# to start printing (0 to 23, default=0)
stoprow - optional, row# to stop printing (0 to 23, default=23, must be>= startrow)

Description

The PSCREEN command "dumps" the contents of screen memory to the printer. This command is useful for many of the BASIC8 commands to obtain a 'hardcopy' of the displayed information.

The PSCREEN command assumes graphics mode 0 (zero), furthermore it assumes that the screen map is the 960 bytes located in memory immediately below PEEK(106)*256 (this is the 'normal' location for the screen map). This command reads 40 bytes of screen memory and sends this to the printer followed with an end of line character (EOL). It then checks to see if it is done printing, if not it repeats the process. The EOL is printed only if location 1009 ($3F1) equals one (initialized to 1 at boot time), if location 1009 is zero no EOL is printed. This is useful if you print to a 40 column printer such as the Atari 820(c).

Example: PSCREEN 0,18

Command : PURGE

Purpose : removes file(s) from a diskette

Form : PURGE filename

Argument : a filename

Description

The PURGE command permanently removes files from a disk. All files matching the file-specifier on the named disk device will be deleted from the disk. The files will no longer be available for any file access nor will they appear when a "DIR" command is issued.

Example: PURGE D1:TEMP

Command : RENAME

Purpose : renames file(s) to a new name

Form : RENAME Dn:oldname,newname

Arguments : Dn - disk name.number
oldname- file-specifier, present name of file(s)
newname- file-specifier, new name for file(s)

Description

The named disk is searched and all occurances of the oldname file-specifier on that diskette are replaced with the newname file-specifier.

CAUTION!

No protection is provided against forming duplicate file-specifiers. Be very careful using wildcards withe the RENAME command.

Example: RENAME D:OBJ.BIN,UTIL.COM

Command : RENUM

Purpose : to renumber BASIC program line numbers

Form : RENUM (begin) (,increment)

Arguments : begin - optional, begin line number for renumber, default= 10
increment- optional, increment between line numbers, default= 10

Description

The RENUM command renumbers the BASIC program in memory, all lines will be renumbered, all internal references will be renumbered also.

RENUM is not able to renumber references to lines which do not exist, or to variable references (ie. GOTO N). When RENUM encounters either of these two conditions the offending line number(s) and the reason for the offense will be displayed to the screen. An example of this is:

displayed:	meaning
100NF	: RENUM could Not Find a line referenced in line#100
260VR	: RENUM found a Variable Reference in line#260

Any line number reference enclosed in parenthesis will be treated as a variable reference. Variable references are not renumbered however there is one exception to this. If a line number reference is a number followed with an operator and an expression, in this case the number WILL be renumbered and the 'normal' error message for variable references will be generated.

NOTE!

During the process of RENUMbering an unresolved line number reference may have become resolved (for instance: if a one line program which consists of "10 GOTO 100" is RENUMbered with the command "RENUM 100" the program then becomes "100 GOTO 100" and the unresolved referenced has become resolved).

CAUTION!

Do NOT strike SYSTEM RESET while RENUMbering. If this is done RENUM may have been interrupted before it was finished and unresolved line number references may have been created.

Example: RENUM
or: RENUM 1000,100

Command : SOFT

Purpose : loads (or saves) a SOFTKEY data file (disk/cassette)

Form : SOFT filename (,SAVE)

Arguments : a filename
SAVE - optional, indicates "SAVE" softkey file, default= "LOAD"

Description

SOFTKEY files may be loaded directly into memory, or saved to disk or cassette files with this command. The files must be compatible SOFTKEY files, that is:

1. the first two bytes must be $53, $53
2. the next four bytes must be the start, end address to load into memory (the end address becomes LOMEM)
3. the remainder of the file is the binary image of the SOFTKEY data (first 64 bytes=lengths, remainder=literals)

SOFTKEY files may be loaded from either cassette or disk, only single step binary loads are supported. The number of bytes transfered and the starting address (in decimal , hex form) will be displayed.

Note, please refer to the SOFTKEY information on page 6 of this booklet for using SOFTKEYS.

Example: SOFT D:KEYS,SAVE
or: SOFT D:KEYS
Count=94 , address=12642 , $3122

Command : STOR

Purpose : to store number(s) or a string directly to memory

Form : STOR startaddress map

Arguments : startaddress- the address in memory to begin storing number(s) or character(s)
map - numbers (separated by commas if more than one) or a string beginning with a quotation mark

Description

The STOR command stores data directly to memory. If the map is a string (begins with a quotation mark) then the characters will be mapped to memory exactly as typed. If the map is a number (or numbers) then the binary representation of that number will be stored at 'startaddress'.

Example: STOR $400 "Copyright 1983 M. D. Pelletier"
or: STOR $421 $9B,155

Command : UNLOCK

Purpose : removes the 'lock status' protection caused by the LOCK command

Form : UNLOCK filename

Argument : a filename

Description

The UNLOCK command removes the write protection caused by the LOCK command so that disk files may be erased, renamed or written to. Any file matching the file-specifier on the named disk will be affected.

Example: UNLOCK D:UTIL.*

Command : VAR

Purpose : displays active variable names, and total count

Form : VAR

Arguments : (none)

Description

The VAR command displays all active variable names as well as the quantity of variables. The sequence of the displayed variables is the sequence in the variable value table.

Example: VAR
N,I,A$,B(
Count=4

Error Codes

NUMBER	MEANING
30	missing required parameter
31	range empty (list or address range)
32	--(not used)--
33	incorrect file type (BLOAD fails)
34	lines won't fit in sequence at new location, COPY fails
35	attempt to copy to line within copy range, COPY fails